

Applying Key Operators in the Observable Class: Case Study ex1

Douglas C. Schmidt

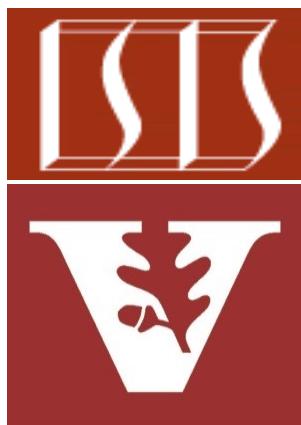
d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

Institute for Software
Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA



Learning Objectives in this Part of the Lesson

- Case study ex1 explores calls to Observable just(), fromArray(), fromCallable(), fromIterable(), map() doOnNext(), mergeWith(), repeat(), & blockingSubscribe() to create, reduce, multiply, & display Big Fraction objects synchronously

Observable

```
.just(BigFraction.valueOf(100,3),  
      BigFraction.valueOf(100,4),  
      BigFraction.valueOf(100,2),  
      BigFraction.valueOf(100,1))  
.map(fraction -> fraction  
      .multiply(sBigReducedFraction))  
.blockingSubscribe  
(fraction -> sb.append(" = "  
      + fraction.toMixedString()  
      + "\n"),  
 error -> sb.append("error")),  
) -> BigFractionUtils  
      .display(sb.toString())));
```

Applying Key Operators in the Observable Class: Case Study ex1

Applying Key Operators in the Observable Class to ex1

The screenshot shows an IDE interface with the following details:

- File Menu:** File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, Help.
- Project:** observable-ex1
- Source Tree:** observable-ex1 / main / src/main / java / utils / AsyncTester, BigFraction, BigFractionUtil, ex1, ObservableEx.
- Open Files:** BigFractionUtil.java, ObservableEx.java (active), ex1.java.
- Code Content (ObservableEx.java):**

```
8 /**
9  * This class shows how to apply RxJava features synchronously to
10 * perform basic Observable operations, including fromCallable(),
11 * repeat(), just(), map(), mergeWith(), and blockingSubscribe().
12 */
13 public class ObservableEx {
14     /**
15      * Test BigFraction multiplication using a synchronous Observable
16      * stream.
17     */
18     public static Completable testFractionMultiplication1() {
19         StringBuilder sb =
20             new StringBuilder(">> Calling testFractionMultiplication1()\n");
21
22         Observable
23             // Use just() to generate a stream of big fractions.
24             // http://reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/Observable.html#just-T-T-T-T-
25             .just(BigFraction.valueOf( numerator: 100, denominator: 3),
26                   BigFraction.valueOf( numerator: 100, denominator: 4),
27                   BigFraction.valueOf( numerator: 100, denominator: 2),
28                   BigFraction.valueOf( numerator: 100, denominator: 1))
29
30             // Use map() to multiply each element in the stream by a
31             // constant.
32             // http://reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/Observable.html#map-io.reactivex.functions.Function-
33             .map(fraction -> {
```

- Bottom Bar:** Git, TODO, Run, Problems, Terminal, Event Log.

See github.com/douglasraigschmidt/LiveLessons/tree/master/Reactive/Observable/ex1

End of Applying Key Operators in the Observable Class: Case Study ex1